Distributional Clauses Particle Filter

Davide Nitti[1], Tinne De Laet [2], and Luc De Raedt [1]

[1]Department of Computer Science, KU Leuven, Belgium.
[2]Tutorial services, Faculty of Engineering Science, KU Leuven, Belgium.
{davide.nitti,luc.deraedt}@cs.kuleuven.be {tinne.delaet}@kuleuven.be

Abstract. We review the Distributional Clauses Particle Filter (DCPF), a statistical relational framework for inference in hybrid domains over time such as vision and robotics. Applications in these domains are challenging for statistical relational learning as they require dealing with continuous distributions and dynamics in real-time. The framework addresses these issues, it supports the online learning of parameters and it was tested in several tracking scenarios with good results.

Keywords: statistical relational learning, probabilistic programming, particle filters, sequential monte carlo, tracking

1 Introduction

Robotics and vision have made a lot of progress in state estimation, planning and learning, often employing probabilistic techniques [7]. However, the majority of the techniques used in these domains cannot easily represent relational information, i.e., objects, properties and the relations that hold between them. This calls for the use of probabilistic programming and statistical relational learning techniques (SRL) [1], which have integrated rich relational representations with uncertainty reasoning. Even though many such formalisms are described in the literature, only few of them have been applied to robotics or vision, especially in an online setting. The main challenges are dealing with the dynamics of the environment, continuous distributions and the real-time aspect. This paper reviews the Distributional Clauses Particle Filter (DCPF) framework [5, 6] that addresses these issues and that has been applied in [6, 3, 4].

2 The Probabilistic Language: Distributional Clauses

The DCPF is based on a dynamic variation of Distributional Clauses [2], a language that extends logic programming formalism to define random variables. A *distributional clause* is of the form $\mathtt{h} \sim \mathcal{D} \leftarrow \mathtt{b}_1, \ldots, \mathtt{b}_n$. Informally speaking, whenever the conditions in the body $\mathtt{b}_1, \ldots, \mathtt{b}_n$ hold, a random variable $\mathtt{h}$ is defined with distribution $\mathcal{D}$. A distributional clause is a powerful template to define conditional probabilities; indeed $\mathtt{b}_i$, $\mathtt{h}$, and $\mathcal{D}$ can contain logical variables

that parametrize the clause. Consider the following examples:

$$\mathtt{n} \sim \mathtt{poisson}(6). \tag{1}$$

$$\mathtt{pos}(\mathtt{P}) \sim \mathtt{uniform}(1, 10) \leftarrow \mathtt{between}(1, \simeq(\mathtt{n}), \mathtt{P}). \tag{2}$$

$$\mathtt{type}(\mathtt{A}) \sim \mathtt{uniform}([\mathtt{magnet}, \mathtt{ferromagnetic}, \mathtt{nonmagnetic}]) \leftarrow \mathtt{object}(\mathtt{A}). \tag{3}$$

Clause (1) states that the number of people $\mathtt{n}$ is governed by a Poisson distribution with mean 6; clause (2) models the position $\mathtt{pos}(\mathtt{P})$ as a continuous random variable uniformly distributed from 1 to 10, for each person $\mathtt{P}$ such that $\mathtt{between}(1, \simeq(\mathtt{n}), \mathtt{P})$ succeeds (i.e., $1 \leq \mathtt{P} \leq \mathtt{n}$ with $\mathtt{P}$ integer). Thus, if the outcome of $\mathtt{n}$ is 2, there will be 2 independent random variables $\mathtt{pos}(1)$ and $\mathtt{pos}(2)$. The term $\simeq(d)$ represents the value of the random variable d. Finally clause (3) describe a uniform distribution over 3 possible types for each object $\mathtt{A}$.

Dynamic Distributional Clauses (DDC) extend Distributional Clauses towards temporal domains. They define a discrete-time stochastic process following the same idea of a Dynamic Bayesian Network. We need clauses that define: 1) the prior distribution: $\mathtt{h}_0 \sim \mathcal{D} \leftarrow \mathtt{body}_0$, 2) the state transition model: $\mathtt{h}_{t+1} \sim \mathcal{D} \leftarrow \mathtt{body}_t$, 3) the measurement probability: $\mathtt{h}_{t+1} \sim \mathcal{D} \leftarrow \mathtt{body}_{t+1}$, and finally, 4) clauses that define a random variable at time t from other variables at the same time: $\mathtt{h}_t \sim \mathcal{D} \leftarrow \mathtt{body}_t$. For example, to describe that the next position of every ball is equal to the current position plus gaussian noise we write:

$$\mathtt{pos}(\mathtt{A})_{t+1} \sim \mathtt{gaussian}(\simeq(\mathtt{pos}(\mathtt{A})_t), cov) \leftarrow \mathtt{ball}(\mathtt{A}). \tag{4}$$

3 Inference and Parameter Learning in DCPF

Given a set of DDC clauses, the DCPF performs filtering, that is, it estimates the current (non-directly observable) world state through the observations obtained from sensors. Formally, filtering or state estimation computes the probability density function $p(x_t|z_{1:t}, u_{1:t})$, where x_t is the current state, $z_{1:t}$ is the set of observations, and $u_{1:t}$ the actions (inputs) performed from time step 1 to t.

Given a model defined as a set of DDC clauses, DCPF performs inference based on particle filtering [7], a Monte-Carlo technique to perform filtering in temporal models. Thus, DCPF is a relational particle filter where each particle $x_t^{(i)}$ is an interpretation, i.e., a set of ground facts for the predicates and values of random variables at time t. A key advantage of the DCPF is that it exploits the relational representation to optimize inference. Rather than working with full interpretations (that list the values for all state variables), DCPF propagates *partial* interpretations (Fig. 1), these are *partial* world descriptions in which the many state variables have been marginalized. This significantly improves the performance with respect to classical particle filters that keep the full state [5].

DCPF supports online parameter learning, that is, state estimation of static variables. Learning can be considered as a state estimation problem, adding the parameters to learn in the state. However, this solution produces poor results due to the degeneracy problem in particle filters. To solve the problem we focused on two simple techniques that have a limited computational cost: artificial dynamics and a variation of resample-move. Details can be found in [6].

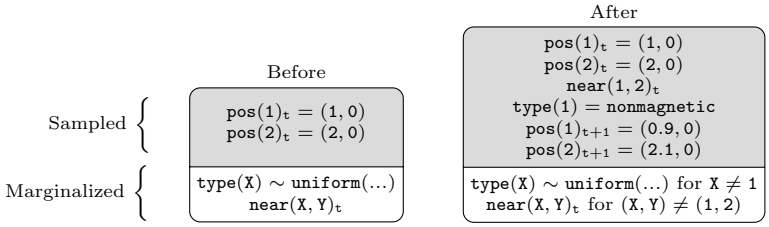


Fig. 1: Partial particle example in the magnetic scenario, before (left) and after (right) the propagation step. Initially the particle contains only the position of the two objects of interest, marginalizing over all other variables. The model states that distant objects do not interact, while close objects interact according to their types. Thus, to sample the next position, DCPF needs to check whether the objects are close. This is the case, so DCPF needs to sample the type of the objects to determine the possible interaction. Object 1 is nonmagnetic in this example, therefore there is no interaction and we can sample the next positions without sampling the type of the second object.

4 Applications

The DCPF framework has been applied to several tracking scenarios [1]. In all scenarios the objects are marked so that their position and orientation can be easily recognized.

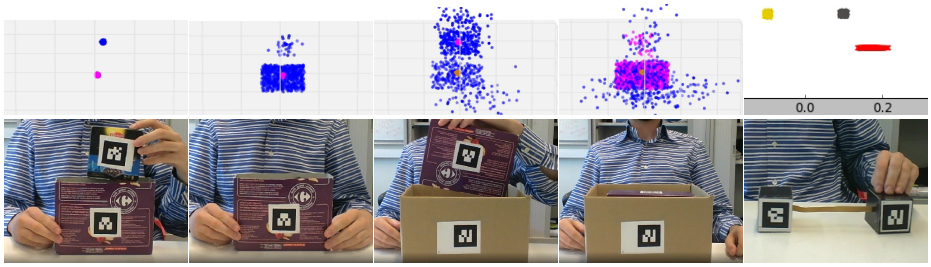
Magnetism scenario [5]: there is a table with objects that can be either permanent magnets, ferromagnetic, or non-magnetic objects. The goal is to track the objects and estimate their type from interactions of pair of objects. To reason about the types of the objects, a theory of magnetism is provided. At the high level it describes interactions, e.g., that two magnets attract or repulse each other. At the lower level, it describes how the positions of the objects evolve over time given the interactions between them.

Box scenario [6]: the goal of this scenario is to track objects moved by a human during a packaging activity with boxes (Fig. 2a-d). The model provided implements principles such as: an object may fall inside the box if it is on the box in the previous step; if the box is rotated upside down the objects inside will fall down with a certain probability and so on. The framework is able to keep track of objects inside boxes, even objects inside a box inside another box.

String scenario [6]: we have a table with several objects possibly connected by strings (Fig. 2e). The goal is to track the objects, estimate the current object directly moved by human and learn online the length of the strings between objects. To perform inference and learning we provide a model in DDC that describe the behavior of objects connected by a string.

Distributional Clauses (the static version) have also been used for modeling affordances in manipulation tasks [3] and occluded object search [4].

[1] Videos available at `https://dtai.cs.kuleuven.be/ml/systems/dc/`



(a) cube on the box (b) cube inside the box (c) rotated box on a beige box (d) cube and box inside the beige box (e) String scenario

Fig. 2: (a-d) packaging scenario. The bottom images represent moments of the experiment, while the top images show the corresponding estimated objects' positions, where each colored point represents an object in a particle. The cube is in blue, the small box in fuchsia and the big box in beige (e) string scenario. The top figure represents the estimated objects' positions (yellow and grey), and the estimated string length in red.

5 Experiments and Conclusions

We proposed a flexible representation for hybrid relational domains in temporal models and provided an efficient inference algorithm for filtering and online learning. This framework exploits the relational representation and the (in)dependence assumptions to reduce the particle size (through partial interpretations) and the inference cost. DCPF is particularly suited for (probabilistic) relational models that involve objects and relations between them. It was empirically evaluated and applied in several tracking scenarios with good results. The results show that DCPF outperforms the classical particle filter, and is promising for more complex robotics applications. The code, papers and videos (of all these scenario's) are available at `https://dtai.cs.kuleuven.be/ml/systems/dc/`.

References

1. De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S. (eds.): Probabilistic Inductive Logic Programming, Theory and Applications. Springer (2008)
2. Gutmann, B., Thon, I., Kimmig, A., Bruynooghe, M., De Raedt, L.: The magic of logical inference in probabilistic programming. Theory and Practice of Logic Programming (2011)
3. Moldovan, B., De Raedt, L.: Learning relational affordance models for two-arm robots. In: International Conference on Intelligent Robots and Systems (2014)
4. Moldovan, B., De Raedt, L.: Occluded object search by relational affordances. In: IEEE International Conference on Robotics and Automation (ICRA) (2014)
5. Nitti, D., De Laet, T., De Raedt, L.: A particle filter for hybrid relational domains. In: International Conference on Intelligent Robots and Systems (IROS) (2013)
6. Nitti, D., De Laet, T., De Raedt, L.: Relational object tracking and learning. In: International Conference on Robotics and Automation (ICRA) (2014)
7. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press (2005)